;login:

The USENIX Association Newsletter

Volume 12, Number 2 **March/April 1987**

CONTENTS

The closing date for submissions for the next issue of *;login:* is April 24, 1987

USENIX THE PROFESSIONAL AND TECHNICAL
UNIX® ASSOCIATION

NOTICE

;login: is the official newsletter of the USENIX Association, and is sent free of charge to all members of the Association.

The USENIX Association is an organization of AT&T licensees, sub-licensees, and other persons formed for the purpose of exchanging information and ideas about UNIX† and similar operating systems and the C programming language. It is a non-profit corporation incorporated under the laws of the State of Delaware. The officers of the Association are:

President	Alan G. Nemeth
Vice-President	Deborah K. Scherrer
Secretary	Waldo M. Wedel
Treasurer	Stephen C. Johnson
Directors	Rob Kolstad Marshall Kirk McKusick John S. Quarterman David A. Yost
Executive Director	Peter H. Salus

The editorial staff of *;login:* is:

Editor and Publisher	Peter H. Salus usenix!peter
Technical Editor	Kevin Baranski-Walker usenix!kevin
Copy Editor	Michelle Dominijanni {masscomp,usenix}!mmp
Production Editor	Tom Strong usenix!strong

Other staff members are:

Betty J. Madden	Office Manager
Emma Reed	Membership Secretary
Jordan Hayes	Technical Consultant

USENIX Association Office
P.O. Box 7
El Cerrito, CA 94530
(415) 528-8649
{ucbvax,decvax}!usenix!office

Judith F. DesHarnais Conference Coordinator

USENIX Conference Office
P.O. Box 385
Sunset Beach, CA 90742
(213) 592-3243 or 592-1381
{ucbvax,decvax}!usenix!judy

John L. Donnelly Exhibit Manager

USENIX Exhibit Office
Oak Bay Building
4750 Table Mesa Drive
Boulder, CO 80303
(303) 499-2600
{ucbvax,decvax}!usenix!johnd

Contributions Solicited

Members of the UNIX community are encouraged to contribute articles to *;login:*. Contributions may be sent to the editors electronically at the addresses above or through the U.S. mail to the Association office. The USENIX Association reserves the right to edit submitted material.

;login: is produced on UNIX systems using *troff* and a variation of the –me macros. We appreciate receiving your contributions in *n/troff* input format, using any macro package. If you contribute hardcopy articles please send **originals** and leave left and right margins of 1″ and a top margin of 1½″ and a bottom margin of 1¼″.

Acknowledgments

The Association uses a VAX‡ 11/730 donated by the Digital Equipment Corporation for support of office and membership functions, preparation of *;login:*, and other Association activities. It runs 4.3BSD, which was contributed, installed, and is maintained by mt Xinu. The VAX uses a sixteen-line VMZ-32 terminal multiplexor donated by Able Computer of Irvine, California.

Connected to the VAX is a QMS Lasergrafix* 800 Printer System donated by Quality Micro Systems of Mobile, Alabama. It is used for general printing and draft production of *;login:* with *ditroff* software provided by mt Xinu.

---

†UNIX is a registered trademark of AT&T.

‡VAX is a trademark of Digital Equipment Corporation.
*Lasergrafix is a trademark of Quality Micro Systems.

MINIX: A UNIX Clone with Source Code for the IBM PC

Andrew S. Tanenbaum
Dept. of Mathematics and Computer Science
Vrije Universiteit
Amsterdam, The Netherlands
Usenet: minix@cs.vu.nl

ABSTRACT

This article describes a new operating system, called MINIX, that is functionally compatible with Version 7 UNIX®. It has been rewritten completely from scratch. Neither the kernel nor the utility programs contain any AT&T code, so the source code is free from AT&T licensing restrictions and may be studied by individuals or in a course. The system runs on the IBM PC, XT, or AT, and does not require a hard disk, thus making it possible for individuals to acquire a UNIX-like system for home use at a very low cost. If a hard disk is available, it is fully supported, however.

Internally, MINIX is structured completely differently from UNIX. It is a message passing system on top of which are memory and file servers. User processes can send messages to these servers to have system calls carried out. The paper describes the motivation and intended use of the system, what the distribution contains, and discusses the system architecture in some detail.

Introduction

MINIX is a new operating system for the IBM PC, XT, and AT. Functionally, it is system-call compatible with Version 7 UNIX, but inside it has been rewritten from scratch. It contains no AT&T code at all, neither in the kernel nor in the utilities. Furthermore, the internal structure is also completely different (it is much more modular). The source code is being released without a restrictive licence for the benefit of those people who would like to have access to the source code of a UNIX-like system. This point is discussed in more detail at the end of this paper.

Another feature of this system is that it has been designed to run with inexpensive hardware. Many people whose work involves computers also have a computer at home. While many of these people have an ego that says "VAX" their wallets often say "IBM PC." MINIX has therefore been designed to run on a 256K IBM PC with one floppy disk if it has to, but the system is then highly restricted. On a 640K IBM PC with two floppy disks, it runs quite well and can even recompile itself from the source code provided, using its own C compiler. On a hard disk system or a PC-AT it works even better, of course. It also runs on those clones that are 100% hardware compatible with the PC, XT, or AT. Experiments have shown that about 80% of all clones are compatible, but 20% are not.

As a natural consequence of this "small is beautiful" philosophy, I decided to have MINIX be compatible with Version 7 rather than, say, 4.3 BSD or System V. Making a 4.3 BSD or System V compatible system that runs on a 256K IBM PC with one 360K floppy disk is left as an exercise for the reader. Besides, there are many people who believe that Version 7 was not only an improvement on all its predecessors, but also on all its successors, certainly in terms of simplicity, coherence, and elegance.

MINIX implements all the V7 system calls, except ACCT, LOCK, MPX, NICE, PHYS, PKON, PKOFF, PROFIL, and PTRACE. The other system calls are all implemented in full, and are exactly compatible with V7. In particular, FORK and EXEC are fully implemented, so MINIX can be configured as a normal multiprogramming system, with several background jobs running at the same time (memory permitting), and even multiple users.

The MINIX shell is compatible with the V7 (Bourne) shell, so to the user at the terminal, running MINIX looks and feels like running UNIX. Over 60 utility programs are part of the software distribution, including `ar`, `basename`, `cat`, `cc`, `chmem`, `chmod`, `chown`, `cmp`, `comm`, `cp`, `date`, `dd`, `df`, `echo`, `grep`, `head`, `kill`, `ln`, `login`, `lpr`, `ls`, `make`, `mkdir`, `mkfs`, `mknod`, `mount`, `mv`, `od`, `passwd`, `pr`, `pwd`, `rev`, `rm`, `rmdir`, `roff`, `sh`, `size`, `sleep`, `sort`, `split`, `stty`, `su`, `sum`, `sync`, `tail`, `tar`, `tee`, `time`, `touch`, `tr`, `umount`, `uniq`, `update`, and `wc`. A full-screen editor loosely inspired by emacs, a full Kernighan and Ritchie compatible C compiler, and programs to read and write MS-DOS diskettes are also included. All of the sources of the operating system and these utilities, except the C compiler source (which is quite large and is available separately), are included in the software package.

In addition to the above utilities, over 100 library procedures, including `stdio`, are provided, again with the full source code.

To reiterate what was said above, all of this software is completely new. Not a single line of it is taken from, or even based on the AT&T code. I personally wrote from scratch the entire operating system and some of the utilities. This took about three years. My students and some other generous people wrote the rest. The C compiler is derived from the Amsterdam Compiler Kit (*CACM*, Sept. 1983), and was written at the Vrije Universiteit. It is a top-down, recursive descent compiler written in a compiler writing language called LLGEN and is not based on or in any way related to the AT&T portable C compiler, which is a bottom-up, LALR compiler written in `yacc`.

Overview of the MINIX System Architecture

UNIX is organized as a single executable program that is loaded into memory at system boot time and then run. MINIX is structured in a much more modular way, as a collection of processes that communicate with each other and with user processes by sending and receiving messages. There are separate processes for the memory manager, the file system, for each device driver, and for certain other system functions. This structure enforces a better interface between the pieces. The file system cannot, for example, accidentally change the memory manager's tables because the file system and memory manager each have their own private address spaces.

These system processes are each full-fledged processes, with their own memory allocation, process table entry and state. They can be run, blocked, and send messages, just as the user processes. In fact, the memory manager and file system each run in user space as ordinary processes. The device drivers are all linked together with the kernel into the same binary program, but they communicate with each other and with the other processes by message passing.

When the system is compiled, four binary programs are independently created: the kernel (including the driver processes), the memory manager, the file system, and `init` (which reads */etc/ttys* and forks off the login processes). In other words, compiling the system results in four distinct *a.out* files. When the system is booted, all four of these are read into memory from the boot diskette.

It is possible, and in fact, normal, to modify, recompile, and relink, say, the file system, without having to relink the other three pieces. This design provides a high degree of modularity by dividing the system up into independent pieces, each with a well-defined function and interface to the other pieces. The pieces communicate by sending and receiving messages.

The various processes are structured in four layers:

4. The user processes (top layer).
3. The server processes (memory manager and file system).
2. The device drivers, one process per device.
1. Process and message handling (bottom layer).

Let us now briefly summarize the function of each layer.

Layer 1 is concerned with doing process management including CPU scheduling and interprocess communication. When a process does a SEND or RECEIVE, it traps to the kernel, which then tries to execute the

command. If the command cannot be executed (e.g., a process does a RECEIVE and there are no messages waiting for it), the caller is blocked until the command can be executed, at which time the process is reactivated. When an interrupt occurs, layer 1 converts it into a message to the appropriate device driver, which will normally be blocked waiting for it. The decision about which process to run when is also made in layer 1. A priority algorithm is used, giving device drivers higher priority over ordinary user processes, for example.

Layer 2 contains the device drivers, one process per major device. These processes are part of the kernel's address space because they must run in kernel mode to access I/O device registers and execute I/O instructions. Although the IBM PC does not have user mode/kernel mode, most other machines do, so this decision has been made with an eye toward the future. To distinguish the processes within the kernel from those in user space, the kernel processes are called **tasks**.

Layer 3 contains only two processes, the memory manager and the file system. They are both structured as **servers**, with the user processes as **clients**. When a user process (i.e., a client) wants to execute a system call, it calls, for example, the library procedure `read` with the file descriptor, buffer, and count. The library procedure builds a message containing the system call number and the parameters and sends it to the file system. The client then blocks waiting for a reply. When the file system receives the message, it carries it out and sends back a reply containing the number of bytes read or the error code. The library procedure gets the reply and returns the result to the caller in the usual way. The user is completely unaware of what is going on here, making it easy to replace the local file system with a remote one.

Layer 4 contains the user programs. When the system comes up, `init` forks off `login` processes, which then wait for input. On a successful login, the shell is executed. Processes can fork, resulting in a tree of processes, with `init` at the root. When Control-D is typed to a shell, it exits, and `init` replaces the shell with another `login` process.

Layer 1 — Processes and Messages

The two basic concepts on which MINIX is built are processes and messages. A process is an independently schedulable entity with its own process table entry. A message is a structure containing the sender's process number, a message type field, and a variable part (a union) containing the parameters or reply codes of the message. Message size is fixed, depending on how big the union happens to be on the machine in question. On the IBM PC it is 24 bytes.

Three kernel calls are provided:

- `RECEIVE(source, &message)`
- `SEND(destination, &message)`
- `SENDREC(process, &message)`

These are the only true system calls (i.e., traps to the kernel). RECEIVE announces the willingness of the caller to accept a message from a specified process, or ANY, if the RECEIVER will accept any message. (From here on, "process" also includes the tasks.) If no message is available, the receiving process is blocked. SEND attempts to transmit a message to the destination process. If the destination process is currently blocked trying to receive from the sender, the kernel copies the message from the sender's buffer to the receiver's buffer, and then marks them both as runnable. If the receiver is not waiting for a message from the sender, the sender is blocked.

The SENDREC primitive combines the functions of the other two. It sends a message to the indicated process, and then blocks until a reply has been received. The reply overwrites the original message. User processes use SENDREC to execute system calls by sending messages to the servers and then blocking until the reply arrives.

There are two ways to enter the kernel. One way is by the trap resulting from a process' attempt to send or receive a message. The other way is by an interrupt. When an interrupt occurs, the registers and machine state of the currently running process are saved in its process table entry. Then a general interrupt handler is called with the interrupt number as parameter. This procedure builds a message of type INTERRUPT, copies it to the buffer of the waiting task, marks that task as runnable

(unblocked), and then calls the scheduler to see who to run next.

The scheduler maintains three queues, corresponding to layers 2, 3, and 4, respectively. The driver queue has the highest priority, the server queue has middle priority, and the user queue has lowest priority. The scheduling algorithm is simple: find the highest priority queue that has at least one process on it, and run the first process on that queue. In this way, a clock interrupt will cause a process switch if the file system was running, but not if the disk driver was running. If the disk driver was running, the clock task will be put at the end of the highest priority queue, and run when its turn comes.

In addition to this rule, once every 100 msec, the clock task checks to see if the current process is a user process that has been running for at least 100 msec. If so, that user is removed from the front of the user queue and put on the back. In effect, compute bound user processes are run using a round robin scheduler. Once started, a user process runs until either it blocks trying to send or receive a message, or it has had 100 msec of CPU time. This algorithm is simple, fair, and easy to implement.

Layer 2 — Device Drivers

Like all versions of UNIX for the IBM PC, MINIX does not use the ROM BIOS for input or output because the BIOS does not support interrupts. Suppose a user forks off a compilation in the background and then calls the editor. If the editor tried to read from the terminal using the BIOS, the compilation (and any other background jobs such as printing) would be stopped dead in their tracks waiting for the the next character to be typed. Such behavior may be acceptable in the MS-DOS world, but it certainly is not in the UNIX world. As a result, MINIX contains a complete set of drivers that duplicate the functions of the BIOS. Like the rest of MINIX, these drivers are written in C, not assembly language.

This design has important implications for running MINIX on PC clones. A clone whose hardware is not compatible with the PC down to the chip level, but which tries to hide the differences by making the BIOS calls functionally identical to IBM's will not run an unmodified MINIX because MINIX does not use the BIOS.

Each device driver is a separate process in MINIX. At present, the drivers include the clock driver, terminal driver, various disk drivers (e.g., RAM disk, floppy disk), and printer driver. Each driver has a main loop consisting of three actions:

1. Wait for an incoming message.
2. Perform the request contained in the message.
3. Send a reply message.

Request messages have a standard format, containing the opcode (e.g., READ, WRITE, or IOCTL), the minor device number, the position (e.g., disk block number), the buffer address, the byte count, and the number of the process on whose behalf the work is being done.

As an example of where device drivers fit in, consider what happens when a user wants to read from a file. The user sends a message to the file system. If the file system has the needed data in its buffer cache, they are copied back to the user. Otherwise, the file system sends a message to the disk task requesting that the block be read into a buffer within the file system's address space (in its cache). Users may not send messages to the tasks directly. Only the servers may do this.

MINIX supports a RAM disk. In fact, the RAM disk is always used to hold the root device. When the system is booted, after the operating system has been loaded, the user is instructed to insert the root file system diskette. The file system then sees how big it is, allocates the necessary memory, and copies the diskette to the RAM disk. Other file systems can then be mounted on the root device.

This organization puts important directories such as */bin* and */tmp* on the fastest device, and also makes it easy to work with either floppy disks or hard disks or a mixture of the two by mounting them on */usr* or */user* or elsewhere. In any event, the root device is always in the same place.

In the standard distribution, the RAM disk is about 240K, most of which is full of parts of the C compiler. In the 256K system, a much smaller RAM disk has to be used, which explains why this version has no C compiler:

there is no place to put it. (The */usr* diskette is completely full with the other utility programs and one of the design goals was to make the system run on a 256K PC with one floppy disk.) Users with an unusual configuration such as 256K and three hard disks are free to juggle things around as they see fit.

The terminal driver is compatible with the standard V7 terminal driver. It supports cooked mode, raw mode, and cbreak mode. It also supports several escape sequences, such as cursor positioning and reverse scrolling because the screen editor needs them.

The printer driver copies its input to the printer character for character without modification. It does not even convert line feed to carriage return + line feed. This makes it possible to send escape sequences to graphics printers without the driver messing things up. MINIX does not spool output because floppy disk systems rarely have enough spare disk space for the spooling directory. Instead one normally would print a file *f* by saying

```
lpr <f &
```

to do the printing in the background. The `lpr` program inserts carriage returns, expands tabs, and so on, so it should only be used for straight ASCII files. On hard disk systems, a spooler would not be difficult to write.

Layer 3 — Servers

Layer 3 contains two server processes: the memory manager and the file system. They are both structured in the same way as the device drivers, that is a main loop that accepts requests, performs them, and then replies. We will now look at each of these in turn.

The memory manager's job is to handle those system calls that affect memory allocation, as well as a few others. These include FORK, EXEC, WAIT, KILL, and BRK. The memory model used by MINIX is exceptionally simple in order to accommodate computers without any memory management hardware. When the shell forks off a process, a copy of the shell is made in memory. When the child does an EXEC, the new core image is placed in memory. Thereafter it is never moved. MINIX does not swap or page.

The amount of memory allocated to the process is determined by a field in the header of the executable file. A program, `chmem`, has been provided to manipulate this field. When a process is started, the text segment is set at the very bottom of the allocated memory area, followed by the data and bss. The stack starts at the top of the allocated memory and grows downward. The space between the bottom of the stack and the top of the data segment is available for both segments to grow into as needed. If the two segments meet, the process is killed.

In the past, before paging was invented, all memory allocation schemes worked like this. In the future, when even small microcomputers will use 32-bit CPUs and 1M × 1 bit memory chips, the minimum feasible memory will be 4 megabytes and this allocation scheme will probably become popular again due to its inherent simplicity. Thus the MINIX scheme can be regarded as either hopelessly outdated or amazingly futuristic, as you prefer.

The memory manager keeps track of memory using a list of holes. When new memory is needed, either for FORK or for EXEC, it searches the hole list and takes the first hole that is big enough (first fit). When a process terminates, if it is adjacent to a hole on either side, the process' memory and the hole are merged into a bigger hole.

The file system is really a remote file server that happens to be running on the user's machine. However it is straightforward to convert it into a true network file server. All that needs to be done is change the message interface and provide some way of authenticating requests. (In MINIX, the source field in the incoming message is trustworthy because it is filled in by the kernel.) When running remote, the MINIX file server maintains state information, like RFS and unlike NFS.

The MINIX file system is similar to that of V7 UNIX. The i-node is slightly different, containing only 9 disk addresses instead of 13, and only 1 time instead of 3. These changes reduce the i-node from 64 bytes to 32 bytes, to store more i-nodes per disk block and reduce the size of the in-core i-node table.

Free disk blocks and free inodes are kept track of using bit maps rather than free lists.

The bit maps for the root device and all mounted file systems are kept in memory. When a file grows, the system makes a definite effort to allocate the new block as close as possible to the old ones, to minimize arm motion. Disk storage is not necessarily allocated one block at a time. A minor device can be configured to allocate 2, 4 (or more) contiguous blocks whenever a block is allocated. Although this wastes disk space, these multiblock **zones** improve disk performance by keeping file blocks close together. The standard parameters for MINIX as distributed are 1K blocks and 1K zones (i.e., just 1 block per zone).

MINIX maintains a buffer cache of recently used blocks. A hashing algorithm is used to look up blocks in the cache. When an i-node block, directory block, or other critical block is modified, it is written back to disk immediately. Data blocks are only written back at the next SYNC or when the buffer is needed for something else.

The MINIX directory system and format is identical to that of V7 UNIX. File names are strings of up to 14 characters, and directories can be arbitrarily long.

Layer 4 — User Processes

This layer contains `init`, the shell, the editor, the compiler, the utilities, and all the user processes. These processes may only send messages to the memory manager and the file system, and these servers only accept valid system call requests. Thus the user processes do not perceive MINIX to be a general-purpose message passing system. However, removing the one line of code that checks if the message destination is valid would convert it into a much more general system (but less UNIX-like).

Documentation

Since one of the purposes of MINIX is to provide a system that can be taught in classes and studied individually ample documentation is essential. For this reason I have written a textbook (719 pages) treating both the theory and the practice of operating system design. The bibliographic data is:

Title: *Operating Systems: Design and Implementation*
Author: Andrew S. Tanenbaum
Publisher: Prentice-Hall, Inc.
Publication date: January 1987

The table of contents is as follows:

CHAPTERS
1. Introduction
2. Processes
3. Input/Output
4. Memory Management
5. File Systems
6. Bibliography and Suggested Readings

APPENDICES
A. Introduction to C
B. Introduction to the IBM PC
C. MINIX Users Guide
D. MINIX Implementers Guide
E. MINIX Source Code Listing
F. MINIX Cross Reference Map

The heart of the book is chapters 2-5. Each chapter deals with the indicated topic in the following way. First comes a thorough treatment of the relevant principles (thorough enough to be usable as a university textbook on operating systems). Next comes a general discussion of how the principles have been applied in MINIX. Finally there is a procedure by procedure description of how the relevant part of MINIX works in detail. The source code listing of appendix E contains line numbers, and these line numbers are used throughout the book to pinpoint the code under discussion. The source code itself contains more than 3000 comments, some more than a page long. Studying the principles and seeing how they are applied in a real system gives the reader a better understanding of the subject than either the principles or the code alone would.

Appendices A and B are quickie introductions to C and the IBM PC for readers not familiar with these subjects. Appendix C tells how to boot MINIX, how to use it, and how to shut it down. It also contains all the manual pages for the utility programs. Most important of all, it gives the super-user password.

Appendix D is for people who wish to modify and recompile MINIX. It contains a wealth of nutsy-boltsy information about everything from how to use MS-DOS as a

development system, to what to do when your newly made system refuses to boot.

Appendix E is a full listing of the operating system, all 260 pages of it. The utilities (mercifully) are not listed.

Distribution of the Software

Software distribution is being done by Prentice-Hall. Four packages are available. All four contain the full source code; they differ only in the configuration of the binary supplied. The four packages are:

- 640K IBM PC version
  (eight 360K diskettes)
- 256K IBM PC
  (no C compiler; eight 360K diskettes)
- IBM PC-AT
  (512K minimum; five 1.2M diskettes)
- Industry standard 9-track tape

The 640K version will also run on 512K systems, but it may be necessary to `chmem` parts of the C compiler to make it fit. The tape version, in addition to the MINIX software, also contains a complete IBM PC simulator in C and other software that allows MINIX to be experimented with to some extent on a VAX or other time sharing computer, rather than a bare IBM PC. This option may be useful for courses for which IBM PCs are not available.

The book ($34.95) and the software ($79.95) are being issued separately. The book can be bought in any technical bookstore, or ordered specially. The book contains a postcard that can be sent back to Prentice-Hall to order the software.

A final word about the legal status of the code is in order. The software does not come with a 10-page licensing document that only the Dean of the Harvard Law School understands. However, it *is* protected by copyright. The software is *not* public domain. However, the publisher does not object to a limited amount of copying being done for noncommercial use. In other words, professors may make copies of the system for their students, students may make copies for their professors, and you may make copies for your friends. If you wish to port the software to another computer and then sell it, you need written permission from Prentice-Hall. In general they will be quite reasonable about granting such permission.

A Usenet newsgroup called **comp.os.minix** has been set up. This channel is being used by people wishing to contribute new programs, point out and correct bugs, discuss the problems of porting MINIX to new systems, etc. Although the publisher does not object to the network being used to broadcast a few files that have been improved, it is not intended to publish the full distribution (eight 360K diskettes) this way.

Acknowledgements

I would like to thank the following people for contributing utility programs and advice to the MINIX effort: Martin Atkins, Erik Baalbergen, Charles Forsyth, Richard Gregg, Michiel Huisjes, Patrick van Kleef, Adri Koppes, Paul Ogilvie, Paul Polderman, and Robbert van Renesse. Without their help, the system would have been far less useful than it now is.

Program for the Phoenix Technical Conference

The 1986 Summer Technical Conference will be held in Phoenix, Arizona, June 8-12.

The USENIX Association will once again offer its well-respected tutorial program. Topics to be presented include:

- Introduction to 4.3BSD Internals
- Advanced 4.3 Internals: Data Structures and Algoritms
- Introduction to System V UNIX Internals
- Advanced System V UNIX Internals
- UNIX Networking
- Managing a Local Area Network (LAN)
- RT/AIX PC Distributed Services
- X-Windows
- The Network File System (NFS)
- UNIX System V Remote File Sharing (RFS)
- Software Development Using C and UNIX
- Special Topics in C
- UNIX Device Driver Design (4.2BSD)
- Language Construction Tools on the UNIX System
- Introduction to Computer Graphics

Tutorials will be held on Monday, June 8 and Tuesday, June 9.

The Vendor Exhibit will be open on Tuesday, June 9, through Thursday, June 11.

This Conference will feature "poster sessions" on Wednesday and Thursday afternoons. These sessions will allow people to give short (5 minute) presentations of current work. A one page (strictly enforced limit!) description of your work (to be posted prior to the session) will be required at least 36 hours in advance so that sessions can be organized and announced. Basic A/V equipment (microphone, overhead projector, slide projector) will be available. Poster sessions will be part of the technical program and will be publicized accordingly.

The preliminary schedule for the Technical Conference is as follows.

WEDNESDAY

W1: Introduction and Keynote (Chair: Eric Allman)

Keynote — Why We Have To Make UNIX Invisible
Steve Jobs, NeXT

The Diamond Multimedia System
Terrence Crowley, BB&N Laboratories

W2: Human Interfaces (Chair: Greg Chesson)

An Interactive WYSIWYG Table Editor
Dirk Debaer, University of Antwerp, and *Sharon Murrel,* AT&T Bell Laboratories

PSFIG — A New Ditroff Preprocessor
Ned Batchelder & Trevor Darrell, University of Pennsylvania

An Environment for SGML Document Preparation
Le van Huu, Università Degli Studi di Milano

W3A: Memory Management (Chair: Dennis Ritchie)

A UNIX Interface for Memory Mapped Files Under Mach
Avadis Tevanian, Jr., Richard F. Rashid, Michael W. Young, David B. Golub, Mary R. Thompson, & William Bolosky, Carnegie-Mellon University

A Replacement for Berkeley Memory Management
Pervaze Akhtar, Gould

Virtual Memory Architecture in SunOS
Robert A. Gingell, Joseph P. Moran & William A. Shannon, Sun Microsystems, Inc.

W3B: Real World (Chair: Dave Taylor)

CAS Perspective on the Maturation of UNIX
Susan A. Funk, Chemical Abstracts Service

Keeping Watch Over the Flocks at Night (and Day)
Kenneth Ingham, University of New Mexico

X.400 on UNIX
Andrew Draskoy & Gerald Neufeld, University of British Columbia

W4A: Programming Systems (Chair: Chris Torek)

The X Toolkit — The Standard Toolkit for X Version 11
Ram Rao & Smokey Wallace, DEC

Shared Libraries in SunOS
Robert A. Gingell, Meng Lee, Xuong T. Dang, & Mary S. Weeks, Sun Microsystems

A Debugger-based System for Graphical Display and Editing of Data Structures
Peter Potrebic & Phil Goldman, Apple Computer

W4B: Poster Session (Chair: Debbie Scherrer)

THURSDAY

T1: Process Models (Chair: Tom Ferrin)

A New IPC System for Bitmap Graphics Applications: Review, Model, Application, and Benchmarks
C. D. Blewett, M. Wish & J. I. Helfman, AT&T Bell Laboratories

Mach Threads and the UNIX Kernel: the Battle for Control
Avadis Tevanian, Jr., Richard F. Rashid, David B. Golub, David L. Black, & Michael W. Young, Carnegie-Mellon University

A Dynamically Extensible Streams Implementation
James Rees, Margaret Olson & J. Sasidhar, Apollo Computer

T2A: Security (Chair: Evi Nemeth)

Rule Based Analysis of UNIX Security
Robert W. Baldwin

UNIX Without the Superuser
M. S. Hecht, C. S. Chandersekaran, R. S. Chapman, L. J. Dotterer, V. D. Gligor, W. D. Jiang, A. Johri, G. L. Luckenbaugh, & N. Vasudevan, IBM

A Partial Model for a B-Level UNIX
Frank Knowles, Gould

T2B: Architecture (Chair: John Mashey)

Protocol Engine Design
Greg Chesson, Silicon Graphics

Virtual Address Cache in UNIX
Ray Cheng, Sun Microsystems

UNIX on a VLIW
Patrick Clancy, Benjamin F. Cutler, Chris Dodd, Douglas Gilmore, Robert P. Nix, & Chris Ryland, Multiflow Computer, Inc.

T3A: File Systems (Chair: Kirk McKusick)

A Remote-File Cache for RFS
Maurice J. Bach & Mark W. Luppi, AT&T Information Systems, and *Anna S. Melamed,* AT&T Bell Laboratories

RFS in SunOS
Howard Chartock, Sun Microsystems

GFS Revisited -or- How I Lived with Five Different Local File Systems
Matt Koehler, DEC

T3B: Internationalisation (Chair: Joe Kalash)

A Common Reader
Thomas Eric Brunner, SRI

Now, UNIX Talks To Me In My Language
Pascal Beyls, BULL

A UNIX System V Streams TTY Implementation for Multiple Language Processing
Hiromichi Kogure & Rick McGowan, AT&T UNIX Pacific

T4A: Compilers (Chair: Don Seeley)

Automatic Error Recovery in a Fast Parser
Bob Gray, University of Colorado

Cross-Module Optimizations: Its Implementation and Benefits
Mark I. Himelstein, Fred C. Chow & Kevin Enderby, MIPS Computer Systems

RPCC — A Stub Compiler for Sun RPC
Irving Reid, University of Saskatchewan

T4B: Poster Session (Chair: John Quarterman)

FRIDAY

F1: Networks (Chair: Jay Lepreau)

Implementing the Reliable Data Protocol (RDP)
Craig Partridge, BB&N Laboratories

Remote UNIX
Michael J. Litzkow, University of Wisconsin

The Network Computing Architecture and System: An Environment for Developing Distributed Applications
Terence H. Dineen, Paul J. Leach, Nathaniel Mishkin, Joseph N. Pato, & Geoff Wyant, Apollo Computer

F2A: Performance

Solving Performance Problems on Multiprocessor UNIX Systems
T. P. Lee, AT&T Bell Laboratories, and *M. W. Luppi & R. E. Menninger,* AT&T Information Systems

Analysis of Real Time Performance
Dave Plauger, Masscomp

Taking Perforance Evaluation Out of the "Stone" Age
Ken J. McDonell, Monash University

F2B: Panel — UNIX Directions? (Chair: Rob Kolstad)

F3: Grab Bag (Chair: Daniel Klein)

UTek Build Environment
Alan McIvor, Tektronix

MK: A Successor to Make
Andrew Hume, AT&T Bell Laboratories

Miranda — An Advanced Functional Programming System Running Under UNIX
David Turner, University of Kent

Experiences with DREGS
Allan Bricker, Morgan Clark, Tad Lebeck, Barton P. Miller, & Peter Wu, University of Wisconsin

---

Full details will be available in a direct mailing from the USENIX Association Conference Office.

≡ ≡ ≡

Ten Years Ago in UNIX NEWS

Agenda

[Meeting in Urbana, IL, 19-21 May 1977]

Thursday May 19

9:00	Opening announcements
10:00	Who is doing what Session
1:00	Inter-Process Communications
evening	Graphics

Friday May 20

9:00	Languages
10:30	Secure UNIX Systems
1:00	Networking
3:00	Data Base Systems
evening	Closed Session: UNIX internals

Saturday May 21

10:00	Text Processing
12:00	Catch-up Session

During both evenings there will be informal "Counsellor" sessions during which experienced Unix hands will try to answer questions of our less experienced brethren and sisteren.

[*UNIX NEWS* April, 1977]

The DASH Project: Design Issues for Very Large Distributed Systems

David P. Anderson
Domenico Ferrari

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, California 94720

Introduction

A *very large distributed system* (VLDS) is a (currently hypothetical) system that:

- is large in the following senses: *numerical* (it contains thousands or millions of connected hosts), *geographical* (hosts may be thousands of miles apart), and *administrative* (the system encompasses hosts and networks owned by many organizations and individuals).
- offers access to non-local resources such as databases, processing power, software, and communication with remote human users.
- is *transparent* in the senses that 1) at some level (perhaps the user interface) the same syntax can be used to access both local and remote resources, and 2) there is little performance difference between local and remote access.

Such a system is preferable to a collection of connected but unintegrated local distributed systems because it allows efficient resource sharing on a much larger scale. However, a VLDS cannot be realized by extending an existing distributed system, because many of the assumptions underlying the designs of these systems do not hold for VLDS. Fundamental differences exist in the areas of security, naming, communication paradigms and architectures, and kernel architecture.

VLDS design involves close interaction of levels ranging from network hardware up to the user programming model, and optimal solutions cannot in general be reached by extending techniques developed for small distributed systems. Furthermore, a VLDS has significant advantages over unintegrated collections of LAN systems, and a properly-designed VLDS restricted to a LAN is potentially as efficient as a specialized LAN system. For these reasons, VLDS design should be viewed as a distinct and important research area.

Previous projects have considered VLDS-related problems such as scalable nameservers [3,6,7] and file services with many clients [4]. Efforts at large-scale integration of existing centralized systems are described in [8] and [5]. These projects, for the most part, address restricted problems or develop solutions based on technology that will soon be outdated.

In contrast, the DASH project at UC Berkeley is taking a unified approach to VLDS design, and is seeking solutions that will not be made obsolete by foreseeable technology advances. The goals of the DASH project are to 1) project the advances in computing and communication hardware that will make a VLDS feasible, and the software systems and applications that will be possible in a VLDS; 2) identify a set of design principles for VLDS, 3) propose mechanisms based on these principles, and 4) develop a prototype VLDS that can be used to test and compare these mechanisms.

Principles and Research Areas

The following are some of the principles for VLDS design that we have arrived at; a more complete discussion can be found in [2]. The DASH prototype incorporates all of these principles.

- Separate the levels of network communication, execution environment, execution abstraction, and kernel structure, and provide an open framework where possible.
- Use a hybrid naming system using a tree-structured symbolic naming for global permanent entities, and capabilities to communication streams for other entities.

- When possible, put communication functions such as security and interface scheduling at a *host-to-host* rather then *process-to-process* level, and consolidate these functions in a *sub-transport* layer.
- Provide flexible support for stream-oriented communication.
- Provide a service abstraction that allows for replication, local caching and fault-tolerance, but does not directly supply them.
- Support real-time computation and communication at every level.

Our discussion raises a number of research areas that should be investigated before a VLDS is put in place:

- Exploring the limits of size and granularity in distributed computing, and identification of possible bottlenecks (process creation, name resolution, authentication, network latency).
- The design of high-performance servers for distant access, and in particular the use of replication, streaming, and caching.
- Utilization of multiprocessors: how important is kernel parallelism, and how does performance depend on processor scheduling, locking mechanisms, and locking granularity?
- Assessing the usefulness of bundle-like stream mechanisms for long-distance high-performance services.
- Investigation of real-time network performance in terms of its implications for network and sub-transport layer design, its implications for process scheduling, and its possible applications.
- Exploring the limits of network performance with very fast networks, network interfaces, and I/O systems.

Project Status

The DASH project currently consists of 2 faculty members, 2 graduate students, and several undergraduates. The DASH prototype kernel is being implemented on Sun-3 workstations in C++. We are using UNIX as our development environment, and have made a modified version of the UNIX dbx debugger that allows us to do remote symbolic debugging of the DASH kernel running on bare machines. The DASH kernel contains no UNIX code.

At this point (February 1987) the lowest layer of the kernel (processes and message-passing) and of the network communication facility (security mechanism and network drivers) have been completed. We have recently performed a study of the performance of our network security mechanisms, described in [1].

References

[1] D. P. Anderson, D. Ferrari, P. V. Rangan and B. Sartirana, "A Protocol for Secure Communication in Large Distributed Systems," UCB/Computer Science Report No. 87/342, CS Division (EECS Dept.) UC Berkeley, February 1987.

[2] D. P. Anderson, D. Ferrari, P. V. Rangan and S. Tzou, "The DASH Project: Issues in the Design of Very Large Distributed Systems," UCB/Computer Science Report No. 87/338, CS Division (EECS Dept.), UC Berkeley, January 1987.

[3] A. D. Birrell, B. W. Lampson, R. M. Needham and M. D. Schroeder, "A Global Authentication Service without Global Trust," *IEEE Symposium on Security and Privacy*, 1986.

[4] M. Satyanarayanan, J. M. Howard, D. A. Nichols, R. N. Sidebotham, A. Z. Spector and M. J. West, "The ITC Distributed File System: Principles and Design," *Proceedings of the 10th Symposium on Operating System Principles*, Operating Systems Review 19, 5 (December 1985), 35-50.

[5] R. E. Schantz, R. H. Thomas and G. Bono, "The Architecture of the Cronus Distributed Operating System," *Proceedings of the 6th International Conference on Distributed Computing Systems*, May 1986, 250-259.

[6] M. D. Schroeder, A. D. Birrell and R. M. Needham, "Experience with Grapevine: the Growth of a Distributed System," *ACM Transactions on Computer Systems* 2, 1 (February 1984), 3-23.

[7] D. B. Terry, M. Painter, D. W. Riggle and S. Zhou, "The Berkeley Internet Domain Server," *USENIX Summer Conference Proceedings*, June 1984, 23-31.

[8] T. Truscott, B. Warren and K. Moat, "A State-Wide UNIX Distributed Computing System," *Proceedings of the 1986 Summer USENIX Conference*, June 1986, 499-513.

Book Review

The Nutshell Handbooks

(Newton, MA: O'Reilly & Associates, 1986) $7.50 each

Reviewed by Lou Katz

Metron Computerware, Ltd.
Oakland, CA
ucbvax!metron!lou

O'Reilly & Associates has created an ambitious set of small volumes intended to serve as a relatively basic introduction to a number of important UNIX† facilities. Uniform in style and presentation, there are currently seven books:

#1 Learning the UNIX Operating System
Grace Todino and John Strang
#2 Learning the Vi Editor
Linda Lamb
#3 Reading and Writing Termcap Entries
John Strang
#4 Programming with Curses
John Strang
#5 Managing UUCP and USENET
Grace Todino and Tim O'Reilly
#6 Using UUCP and USENET
Grace Todino
#7 Managing Projects with Make
Steve Talbott

The first book of this series, *Learning the UNIX Operating System,* declares its intention to give a "good overview of ... UNIX survival materials for the new user," "not to overwhelm you with unnecessary details but to make you comfortable as soon as possible in the UNIX environment." In that respect, the book accomplishes its aim. It is well-organized and flows in a logical manner. The text is simply written, and should make a novice user comfortable. I like the small (8½" × 5½") format, which is compatible with many of the commonly available UNIX manuals.

However (now for the bad news), this volume is seriously flawed in detail. I get the very strong impression that its authors are new to UNIX, mastered the one system they had access to, and by virtue of being more facile than their friends came to delude themselves that they were experts. This seems clear to me by omission, for surely anyone with any depth of experience would have gone to some pains to remark on the existence of different versions of UNIX and to specify (at least in an introduction or appendix) which version or versions this volume referred to. Although one can smile with wry amusement at the statement "`ed` was first developed when text editing was done on a line printer," one wonders whether the authors were born yesterday or just arrived from Mars. Statements of this sort certainly do not inspire confidence! In fact, this book is loaded with inaccuracies in details or concepts, flaws which can easily lead a new or naive user to formulate a fundamentally incorrect model of the system. There is a blurring and mixing of the distinction between UNIX the operating system, the shell user interface and other user-level commands which are supplied with a UNIX system. The assertion that "If you make a mistake in specifying the options (to a command) UNIX (sic!) will display the correct form" is both conceptually and factually incorrect. **Some** commands will give a "usage: syntax" response when faced with unknown flags or missing but required arguments, but many commands do not do so, and certainly **UNIX** isn't doing this service.

Almost all topics are explained in narrative and then at least one specific example is presented. This is very useful, and is reasonably well done, though there are enough errors in detail to make me worry. The overall presentation of I/O redirection is good and there is an excellent discussion of the danger of `qrm`. The discussion of `cd` has a good comment regarding the fact that you cannot `cd` to a filename, but follows that with an example which shows a system response which is incorrect. The description of `ls -a`

† UNIX is a registered trademark of AT&T. All sorts of other things are trademarks of other companies.

poses the common quesion "What are these files '.' and '..'?" explicitly, but then doesn't answer it. Furthermore, the parent directory of the user's directory is shown in an `ls -l` listing as being owned by the user himself, a highly unlikely organization of file system ownership.

Users are encouraged to try a terminal's BREAK key, among others, if they need the INTERRUPT function, a practice which usually will not work, and will often cause big problems. The much-needed section on what to do if your terminal seems hung or unresponsive contains a number of excellent suggestions, but omits crucial ones: to try pressing LINE-FEED or control-J!

Typographically, O'Reilly & Associates have chosen to use a rather poor sans serif font when displaying literal computer-user interactions. I would have preferred greatly the computer's prompts and responses to be bold-faced and the user's entries to be normal rather than the reverse scheme used. In the section on the need for white space between commands and their arguments, the example `ls-l` (without whitespace) was broken at the margin right after the `ls`, totally destroying the visual information of the missing space and the intelligibility of the example!

Although this volume is clearly and simply written and covers a very suitable selection of topics for novice users, it is much too flawed. I cannot recommend it.

Volume #2, *Learning the Vi Editor,* covers that common utility. Since I am not a `vi` user, I cannot vouch for its detailed accuracy. However, the discussion of cursor position on page 3 is totally confusing. The WYSIWYG statement on page 6 is wrong, and the table on customization is unclear. Some of the examples in the regular expression section, though correct, were more complicated than was needed: to delete all trailing blanks, they suggested `:g/\(.*\) *$/s//\1/` while the expression `:g/ *$/s///` would work just as well and didn't need the saved sub-string \1. However, the overall description seemed good enough that I expect to try use this book on the next occasion that I have to cope with a system which does not happen to have my favorite editor but does have `vi`.

Volume #3, *Reading and Writing Termcap Entries,* is written for system administrators, albeit ones with limited experience. It proceeds through the features and capabilities of the termcap system and gives expanded explanations in rather clear English. The "Terminal Capabilities by Function" table is a very nice restatement of the capabilities codes. The blow-by-blow annotation of the entries for two different types of terminals is good, as are the hints on how to write, test, and debug entries. The alphabetic list of capabilities at the end also serves as an index to the booklet. In a few places, the descriptions get bogged down — the written discussion of the '%' arguments being an example, though the following examples partly compensate for the confusion in the paragraph. In general this volume is a useful and helpful addition to the standard documentation.

Volume #4, *Programming with Curses,* the companion volume to #3, is a different story. I actually used this book during a recent project which required curses. The expanded explanations were a great help in dealing with the standard CURSES document in the UNIX reference manuals, but the illustrative examples were poor and badly explained. Detailed explanations of some of the functions were either inaccurate or misleading, leading me to believe that the author did not fully understand, among other things, the gory details of terminal I/O.

Discussion of the `clearok()` function was confusing and left me no wiser than before I read it. Physical echo does **not** mean that "characters are also echoed to the screen locally by your terminal." The definition of `crmode()` says that ^S, ^Q, ^C, ^Y go to the kernel for processing long before they are defined in the text as flow control, interrupt and quit, and there is **NO** mention of local variability and freedom in assigning these functions to the control key of your religious choice. "Raw mode is good if you do not want to handle interrupts and quits and would rather ignore them" is a rather bizarre and dangerous view of this topic.

In general, the usage examples are poor. For me, the greatest lack was in not coming to grips with using curses for overlapping windows. This topic was not treated in any useful fashion. Since `curses` is concerned

with the care and feeding of visual material, the lack of illustrations (each potentially worth 1000 words) is regrettable.

The Quick Reference Table at the end of the book did serve as a partial index. Although seriously flawed, this book was somewhat useful to me, and I would recommend it to experienced programmers, but only very cautiously to novices.

Volume #5, *Managing UUCP and USENET,* and volume #6, *Using UUCP and USENET,* are another pair of pamphlets, one targeted towards the system administrator, the other towards a user who may not even be a programmer. I became concerned at the outset with the technical breadth of experience of the author because I found on page 3 of Volume #5 the statement "we use the system prompt # to indicate that the commands can only be executed in the *superuser* mode because of UUCP file permissions. Commands that are preceded by the system prompt `unix%` can be invoked in multi-user mode." Anyone who confuses permissions with operating mode (I have never heard **single-user mode** referred to as **superuser mode**), or who cannot find the right words to express this concept, makes me wary, as does someone who believes the 50 foot limit on direct RS-232 connections. The discussion of null modems is too trivial (again lack of knowledge?), for a new system administrator really needs some discussion of modem control lines, not just how to cross send-data and receive-data wires.

There are many detailed tidbits which seemed strange, such as putting comment lines in */etc/passwd* by starting a line with #, doing a `setuname` on every reboot from */etc/rc*, or putting several entries for the same username but with different passwords in */etc/passwd* and expecting `login` to find the one which matched. `getty` monitors incoming lines, and does not start a `login` when a call out is initiated. Setting the "time to call" field in *L.sys* does not **enable** `uucico`.

The output from `uucico` in debug mode is confusing and not really explained in standard UNIX documentation, and this book adds nothing to that situation. A two page reproduction of a session is dropped in the user's lap without so much as one line of comment, aside from dividing the printout into sections and identifying the major activity underway (establishing contact, sending a file, etc). However, the table of STST and LOGFILE messages is good and useful.

As I have never had to install or maintain `netnews`, I cannot vouch for the accuracy of the chapters on that subject. What I read seemed clear and useful. Appendix A, which lists the names, formats and use of the `uucp` working files is a welcome addition of useful information on this cryptic system. Despite the glaring flaws there is value in this book, and I give it a qualified recommendation.

The companion volume #6 is meant for users, and suffers from the same defects as the other volumes. UUENCODE is **not** a UUCP command for sending binary files, and even the example shows encoding a file and then piping it through mail. One strongly suspects that much of the information that is presented was obtained by reading the documentation rather than by trying it.

The fact that tilde escapes like `~!` or `~%put` had to start on a new line was never mentioned. The author seems not to have known that the `~>:` file diversion facility of `cu` could be invoked without the trailing `:`, and so wastes considerable time explaining how to run scripts to see material, when `~>` enables you to see the input while capturing it in a file. If one never gets past the `~%take` and `~%put` options, one doesn't quite understand the full power of the `cu` program. As with the other booklets, the discussion of `netnews` is quite thorough, but I was tired of looking for defects by then. With the recent reorganization of the entire newsgroup naming conventions, some of the material is out of date, but that isn't the fault of the author. This book is a useful addition to the standard documentation, but don't go to any special trouble to find it.

Volume #7, *Managing Projects with Make,* is a rather comprehensive tour through the `make` facility. Written in an entirely different style from the standard UNIX documentation, it provides a clear and comprehensive description of the flags, options, and features of `make` along with examples. The booklet discusses the less than obvious syntax of `make` and its macros in some detail, and I learned a few things about maintaining libraries using it. My main criticism is that I would have preferred more explanation of why certain

constructs were used in the final rather elaborate example. All in all I would recommend this book, especially for the experienced programmer who may be working alone and without much contact with a `make` guru, and who need to get started using this facility.

In summary, O'Reilly & Associates have tackled many of the functions new or inexperienced users, programmers, and system administrators need to know in a coherent and unified fashion, and should be commended for their efforts. Each book is complete in itself and modestly priced (about $7), though the cost for the entire set is no longer trivial. Each volume contains a Table of Contents at the beginning and Summary pages at the end, but none contains an index. The subjects are covered thoroughly and in detail and the writing is generally clear and simple. This is not just a re-phrasing of material found in the standard UNIX manuals.

However, when examined closely, the books are riddled with inaccurate, misleading, or downright incorrect technical details or concepts, so that their utility is in some cases severely compromised. In a discipline where literal accuracy is required for learning by example, this failing is inexcusable and cannot be taken lightly. The printing style, photo offset onto small format stapled booklets with inexpensive paper should make revision less costly, and many of the problems noted above are correctable. With the exception of volume #1, the other six booklets are all more useful than flawed, and can be recommended to a greater or lesser degree. The table below summarizes my ratings for each of the books. The rating scale is:

Unacceptable	0
Poor	1
Satisfactory	2
Good	3
Excellent	4

	#1	#2	#3	#4	#5	#6	#7
Clarity of Presentation	2	3	3	3	2	3	4
Organization and Style	3	3	3	3	3	3	4
Examples	2	3	3	1	2	2	3
Completeness	3	3	3	2	3	3	3
Accuracy	1	2	3	3	2	2	3
Overall Value	1.5	3	3	2.5	2.5	2.5	3.5

We're Moving!

At the end of March, the USENIX Association will be moving to new office space. The new office will give us just over double the floor space. Our new postal address will be:

P.O. Box 2299
Berkeley, CA 94710

The telephone number will remain 415-528-8649.

Summary of the Board of Directors' Meeting Monterey, October 1-2, 1986

Attendance

Present at the meeting were: Stephen C. Johnson, Rob Kolstad, Alan G. Nemeth (President), John S. Quarterman, Deborah K. Scherrer, Waldo M. Wedel, David A. Yost — Directors; Rick Adams, Washington local host; Daniel L. Appelman, Kadison, Pfaelzer, Woodard, Quinn & Rossi; Judy DesHarnais, Conference Coordinator; John L. Donnelly, Exhibit Manager; Jeannie Patton, Wells Communications; Peter H. Salus, Executive Director.

Manuals

The question of reprinting the 4.2BSD manuals was discussed. It was felt by the Board that as the differences between 4.2 and 4.3 were listed in the new manuals, the Association's commitment to 4.3BSD was such that 4.2BSD manuals should not be reprinted.

Office Matters

Because of the inadequate nature of the Association's office in El Cerrito, the Executive Director was empowered to seek more appropriate space. Yost was empowered to spend up to $35,000 to upgrade the Association's hardware, while still pursuing a possible equipment donation.

Graphics Workshop

DesHarnais reported on the status of the forthcoming 3rd Monterey Computer Graphics Workshop. Though there had been some organizational problems, everything was now in order. The Workshop will be held November 20-21.

Washington Meeting

DesHarnais circulated the advertisement, which was approved without dissent. There was a discussion of expenses for those offering tutorials and of the complimentary admissions policy. It was felt that the tutorial rooms in Washington were ideal. The various programs were discussed and their progress noted. Salus reported on the bus shuttle situation.

Phoenix Meeting

DesHarnais informed the Board that Steve Jobs had accepted the invitation to be Keynote Speaker. Eric Allman is program chair. There was a discussion of possible locations for a reception in Phoenix. There was also discussion of future tutorial topics.

Future Meetings

There was discussion of the format of the Dallas meeting. The question of three one-day miniworkshops was mentioned. The issue of greater variety was brought up. Kolstad and Yost asked about the meeting times (June and January) and the historical basis for this was gone into.

San Diego was selected as the site for the January 1989 meeting. The Winter 1990 and 1991 sites are still open, but the Board decided to wait until more was known about /usr/group's plans. The Board left the question of June 1991 open, as well.

Licensed tutorials

With Dan Appelman, the Board's lawyer, present, there was an extensive discussion concerning admission to licensed tutorials and the Association's responsibility where attendees were concerned. After examining past registration forms, Appelman stated that the USENIX statements appeared to be in good order. He said that were the Association to behave as it had in the past, it would clearly be held harmless. It was concluded that the Association would continue to check license status of attendees.

Software Distribution

The notion of a software distribution project, more extensive than the Association's tape distribution, and not limited to licensed members, was brought up by Yost. In the course of the discussion, the notion of making all of *mod.sources* available arose. Kolstad agreed to find an individual to put *mod.sources* on tape for distribution. It was also agreed to make the USENIX 86.2 tape available to individual members at $100.

Wells Communications

There was a brief discussion of the difference between advertising and public relations, and the need of the Association for a "better image." It was agreed to retain Wells Communications to act on behalf of the Association up to and including the Washington conference.

Technical Editor

It was decided to empower Kolstad and Salus to staff *;login:* as they saw fit. They will post an editorial job description on the net and make a decision.

Stargate

Wedel and Quarterman presented a business plan based by Wedel on the information received from Horton and Weinstein. It appeared to demonstrate that the Stargate proposal was not economically feasible. It was decided to inform Weinstein and Horton that the Association would most likely not continue its support of the Stargate experiment, but that a final decision would be made in January.

— PHS

Summary of the Board of Directors' Meeting Washington, DC, January 19-20 & 22, 1987

Attendance

Present at the meeting were: Stephen C. Johnson, Rob Kolstad, Marshall Kirk McKusick, Alan G. Nemeth (President), John S. Quarterman, Deborah K. Scherrer, Waldo M. Wedel, David A. Yost — Directors; Rick Adams, Washington local host; Eric Allman, Phoenix program chair; Judy DesHarnais, Conference Coordinator; John L. Donnelly, Exhibit Manager; Lou Katz, Past President; Betty Madden, Office Manager; Peter H. Salus, Executive Director

Treasurer's Report

Johnson reported that the 1986 Financial Statements showed the Association to be in good standing. It was unanimously agreed that the Statements be published in this issue of *;login:*. Johnson related that he had consulted with Mike Eggar of Arthur Young, Inc., who had examined the Association's papers and felt that USENIX did not need a formal audit. It was unanimously agreed that there be no audit at this time. It was further agreed that Johnson and Salus consult to improve the Association's office procedures.

New Office Space

Salus reported that he had signed a lease for new office space. The Association will double the square footage available to it in its new space in Berkeley.

4.3BSD Manuals

The new 4.3BSD Manuals were displayed with much relief. Howard Press is beginning shipments. It was reported that half the *User's Sets* had already been sold and that a second printing would most likely be necessary by the end of February.

Phoenix Meeting

Allman reported on the timetable for papers for the June meeting. DesHarnais reported on the hotel arrangements. Donnelly related the exhibit schedule and informed the Board that there were 19 exhibitors so far. After a lengthy discussion it was decided to get a marketing person to aid Donnelly in space sales.

Sources

Kolstad introduced Hokey (representing Plus5 Computing) to the Board as the person who will collect programs from the net for software distribution. Hokey reported that he had over 150Mbytes of sources which he was breaking into 40Mbyte chunks. He said that they were planning to add in README files and man pages.

Future Meetings

The Board decided that the January 1989 meeting would be held at the Town and Country in San Diego. The Board tentatively decided to meet concurrently with UniForum in Washington, DC, in 1990. The 1991 Summer meeting will be held in Nashville.

The Board approved arrangements for the Large Systems Administrators' Workshop (Philadelphia) and the 4th Graphics Workshop (Cambridge, MA).

Stargate

After a lengthy discussion, the Board decided to conclude its financial support of the Stargate experiment at the end of February. The following notice was posted at the Washington conference and on *comp.org.usenix*:

> The USENIX Board of Directors is pleased to announce that the Stargate experiment has demonstrated technical feasibility. The conclusion of this experiment will be funded by the Association through the end of February 1987.
>
> Stargate Information Services, the product of this lengthy experiment, will be organized by the investigators during the next few weeks. Its purpose is to provide news by means of a satellite via cable TV or a dish.
>
> Further information is available from Mark Horton (cbosgd!mark) or Lauren Weinstein (cbosgd!vortex!lauren).

UUNET

Rick Adams and Mike O'Dell presented a proposal for a news and mail distribution system, to be called UUNET, involving public packet-switch networks. The Board was most interested in the proposal and the principals were asked to work with Quarterman and Wedel in order to present a business plan and trial service at the next Board meeting.

mod.std.unix Archives

Quarterman brought up the question of USENIX being the repository of the archives. It was agreed that when new hardware was installed, the Association keep a copy of the archives and distribute them by tape and paper mail. ARPA Internet distribution would still be done by anonymous FTP from sally.utexas.edu.

P1003

Quarterman reported on the POSIX standard meetings. He mentioned that a "rationale" document was needed to supplement the full standard and that he had been approached about writing it. Quarterman wanted to know whether, as this was funded by /usr/group, the Board saw any conflict in his doing so. The consensus was that there was no objection nor conflict.

Nominating Committee

Nemeth announced that he had approached Lew Law concerning chairing the Nominating Committee for the 1988 elections. Law had expressed his willingness and the Board echoed the Presidential appointment.

Wells Communications

It was decided to engage Wells Communications through the Phoenix conference and to enlarge their purview to include periodic press releases to keep the Association before the public eye. A number of topics for such releases was discussed. Salus was designated liaison for the public relations effort.

Face Server

Katz and Yost introduced their proposal to set up a "face server" at the Phoenix conference, so that the attendee list will contain postage-stamp size digitized images of attendees next to their names and addresses. The experiment was approved.

Dallas Conference

Rob Kolstad was unanimously designated program chair for the Dallas Conference, which will be concurrent with UniForum in 1988.

Foreign UNIX Groups

Scherrer spoke to her proposal concerning relations with EUUG, AUUG, and NZUSUGI. It was explained that it was necessary to formalize relations where *;login:*, tapes and manuals were concerned. Scherrer and Salus were asked to take appropriate steps to ensure circulation of the newsletter and of proceedings; Salus was asked to consult with a lawyer concerning sales of the manuals.

Other Business

It was decided to encourage more workshops on a variety of topics.

It was decided to offer a prize for the best student paper at the Phoenix conference.

Membership categories was designated as a topic for the Board meeting in Phoenix.

— PHS

≡ ≡ ≡

Future Meetings

Large Installation System Administration Workshop
April 9 & 10, 1987, Philadelphia, PA

For information contact the USENIX Conference Office at (213) 592-3243 or *usenix!judy*.

4th NZUSUGI Meeting – Auckland

The New Zealand UNIX Systems User Group Inc. will hold its fourth annual conference May 13-16, 1987, at the Travelodge at the Auckland International Airport. The theme of the conference is "UNIX as you like it." There will be pre-conference seminars and both commercial and technical streams.

For further information, contact:

Ian M. Howard
c/o CCL Business Systems Ltd.
PO Box 3323
Auckland, New Zealand

4th Computer Graphics Workshop
Oct. 8 & 9, 1987, Cambridge, MA

Abstracts are due by May 1, 1987. For information, contact Tom Duff, the program chair, at *research!td* or (201) 582-6485.

USENIX 1988 Winter Conference and UniForum – Dallas

The USENIX 1988 Winter Conference will be held on February 10-12, 1988, at the Registry Hotel in Dallas, Texas. It will be concurrent with UniForum 1988, which will also be in Dallas. The Conference will feature tutorials and technical sessions.

USENIX 1988 Summer Conference and Exhibition – San Francisco

The USENIX 1988 Summer Conference and Exhibition will be held on June 21-24, 1988, at the Hilton Hotel in San Francisco, California. There will be a conference, tutorials, and vendor exhibits.

Long-term USENIX Conference Schedule

Jun 8-12 '87 Hyatt Regency, Phoenix, AZ
Feb 10-12 '88 Registry Hotel, Dallas, TX
Jun 21-24 '88 Hilton Hotel, San Francisco, CA
Feb 1- 3 '89 Town & Country Inn, San Diego, CA
Jun 13-16 '89 Hyatt Regency, Baltimore, MD
Jun 11-15 '90 Marriott Hotel, Anaheim, CA

Elliott D. Buchdruker
Certified Public Accountant
Pier 15, The Embarcadero
San Francisco, CA 94111
(415) 362-6204

The Board of Directors
USENIX ASSOCIATION

The accompanying balance sheet of USENIX ASSOCIATION as of November 30, 1986 and the related statements of revenue and expenses and changes in financial position for the year then ended have been compiled by me.

A compilation is limited to presenting in the form of financial statements information that is the representation of management. I have not audited or reviewed the accompanying financial statements and, accordingly, do not express an opinion or any other form of assurance on them.

Elliott D. Buchdruker

January 9, 1987

USENIX ASSOCIATION
BALANCE SHEET
NOVEMBER 30, 1986
(Unaudited)

ASSETS

CURRENT ASSETS:		
Cash in bank:		
Mechanics Bank	$ 67,877	
First Interstate Bank	226,001	
First Interstate Bank, Conference Account	29,011	
Colorado National Bank, Exhibit Account	28,555	$351,444
Marketable securities		217,054
Claim for income tax refund		10,926
Prepaid conference and exhibit costs		33,884
Total current assets		613,308
FIXED ASSETS, AT COST	66,057	
Less - accumulated depreciation	(20,084)	45,973
		$659,281

LIABILITIES AND FUND BALANCE

CURRENT LIABILITIES:		
Deferred revenue		$ 24,035
Payroll taxes payable		1,729
Accrued expense		599
Total current liabilities		26,363
FUND BALANCE:		
Balance, December 1, 1985	$543,207	
Excess of revenue over expenses for the year ended November 30, 1986	89,711	632,918
		$659,281

See accompanying notes to financial statements.

USENIX ASSOCIATION
STATEMENT OF REVENUE AND EXPENSES
FOR THE YEAR ENDED NOVEMBER 30, 1986
(Unaudited)

REVENUE:		
Membership dues		$147,101
Conference income (Exhibit A)		338,019
Proceedings		23,945
Rebates		5,901
Dividend income		13,231
Interest income		25,670
"T" shirts		9,856
Miscellaneous		10,145
		573,868
EXPENSES:		
Program services:		
Newsletter	$ 15,381	
Tape service	9,459	
Proceedings	15,276	
Stargate	1,084	
Other	418	41,618
Supporting services:		
Salaries	108,744	
Payroll taxes	10,009	
Consulting and contract labor	67,554	
Office management fees and service	2,549	
Computer	9,089	
Telephone	18,100	
Meetings and travel	45,502	
Refunds	10,474	
Postage and shipping	32,925	
Depreciation	10,869	
Occupancy	19,011	
Equipment rental, maintenance and repair	6,149	
Printing and typesetting	14,596	
Supplies	5,061	
Staff recruiting and training	1,915	
Boulder office	26,184	
Legal and accounting	6,903	
Insurance and bonds	11,201	
Taxes and licenses	20,684	
Bank charges	3,421	
Advertising and promotion	3,064	
Miscellaneous	8,535	442,539
		484,157
REVENUE OVER EXPENSES		$ 89,711

See accompanying notes to financial statements.

ELLIOTT D. BUCHDRUKER, C.P.A.

USENIX ASSOCIATION
CONFERENCE INCOME - EXHIBIT A
FOR THE YEAR ENDED NOVEMBER 30, 1986
(Unaudited)

	Conferences			
	Atlanta	Monterey	Denver	Total
REVENUE:				
Registration fees	$414,722	$ 21,341	$319,218	$755,281
Exhibitor fees	86,660			86,660
	501,382	21,341	319,218	841,941
EXPENSES:				
Refunds		200	12,465	12,665
Pre-conference and on-site printing and typesetting		918		918
Pre-announcement mailing	6,881		5,171	12,052
Pre-registration	18,709		13,656	32,365
Registration packet	24,915		13,277	38,192
Tutorial expense	76,032		45,492	121,524
Rent	8,820	1,200		10,020
Advertising	10,105		13,076	23,181
Promotion	2,283			2,283
Credit card charges		278	4,925	5,203
Audio visual	8,121	2,554	1,566	12,241
Temporary office help	12,699	3,683	9,529	25,911
Telephone	2,645	949	1,670	5,264
Xerox	1,339		15	1,354
Consulting and planning fees	34,080	13,940	20,000	68,020
Travel	1,680	1,844	3,231	6,755
Entertainment	1,953			1,953
Conference signs	635		520	1,155
Hotels	7,895	3,999	2,174	14,068
Catering and restaurant charges	16,007	647	15,378	32,032
Shipping and postage	3,699	2,432	2,881	9,012
Mailing service fees		909		909
USENIX and /usr/ Group booth			311	311
Security guards	4,560		185	4,745
Receptions	42,000	834		42,834
Office supplies	979	182	962	2,123
Equipment rental	964		182	1,146
Bank charges	6,313		142	6,455
Program committee expenses	4,743	2,277		7,020
Decorations	2,168	43		2,211
	300,225	36,889	166,808	503,922
Conference income	$201,157	$(15,548)	$152,410	$338,019

See accompanying notes to financial statements.

ELLIOTT D. BUCHDRUKER, C.P.A.

USENIX ASSOCIATION
STATEMENT OF CHANGES IN FINANCIAL POSITION
FOR THE YEAR ENDED NOVEMBER 30, 1986
(Unaudited)

SOURCES OF FUNDS:	
Net income	$ 89,711
Add - depreciation not requiring a current outlay of cash	10,869
	$100,580
APPLICATION OF FUNDS:	
Increase in working capital	$ 77,382
Purchase of fixed assets	23,198
	$100,580
INCREASE (DECREASE) IN WORKING CAPITAL COMPONENTS:	
Cash	$ 91,596
Marketable securities	13,231
Prepaid conference and exhibit costs	(18,123)
Deferred revenue	(7,640)
Accrued expenses	(599)
Payroll taxes payable	(1,083)
	$ 77,382

See accompanying notes to financial statements.

USENIX ASSOCIATION
NOTES TO FINANCIAL STATEMENTS
NOVEMBER 30, 1986
(Unaudited)

1. Corporate Background

Usenix Association was incorporated in 1980. The principal purpose of the organization is to provide educational benefits, including the exchange and communication of research and technological ideas pertaining to Unix™ and Unix related computer systems. It is a nonprofit public charity established under Section 501 (c)(3) of the Internal Revenue Code. Contributions to the organization are deductible by the donors.

2. Summary of Significant Accounting Policies

Marketable securities are recorded at cost. Usenix Association uses the direct charge-off method of recognizing doubtful accounts. Property and equipment are recorded at cost. The Company follows the practice of capitalizing all expenditures for property and equipment in excess of $400. Depreciation is calculated according to the accelerated cost recovery system for property placed in service after 1980. The lives are comparable to those used under generally accepted accounting principles. For all other property, depreciation is calculated over the estimated useful lives of the assets on both the straight line and the accelerated basis.

ELLIOTT D. BUCHDRUKER, C.P.A.

WEIRDNIX Competition

The winner in the most serious category was Paul Gootherts.

Problem:

The definition of `sleep()` is inconsistent.

Explanation:

"The value returned by the `sleep()` function shall be the unslept amount (the requested time minus the time actually slept)." [Paragraph 3.4.3.3]

"The suspension time may be longer than requested by an arbitrary amount due to the scheduling of other activity in the system." [Paragraph 3.4.3.2]

Since the time actually slept can be greater than the time requested, the value returned could be negative. However, `sleep()` returns an unsigned `int`. [Paragraph 3.4.3.1]

Proposal:

`sleep()` could be changed to return a `signed int`. This is nice because the process that called it could get some idea of how "late" the alarm came.

Alternatively, the routine could be documented to return zero if the actual time was greater than the requested time.

Paul Gootherts
Hewlett Packard

The winner in the "most demented" category was Michael Gersten.

Ok, let's look at `read()` and `write()`.

1. There is no requirement that anything written will be available for a `read()`.

2. There is no requirement that `read/write` return everything that they can.

In general, you can't require this. The terminal lines are a good example; writing to a terminal will not result in it being readable; the terminal drivers only return a line at a time no matter how much is requested. Or at least, that's what the documentation says (I've never actually tested it, but it seems that if it were false, then type ahead would not work as well).

In general, it is probably safe to require that anything written to a file should be available to a subsequent read provided that the read is done on a file descriptor corresponding to the same name, or a link to the same named file that was written to, all providing that it is a regular file. Certainly not for device or special files.

Incidentally, don't think that 2 is obvious; my first UNIX programs assumed that the O/S would return a number of bytes so that the reads would be re-aligned on a 512 byte boundary, and that I had to call `read()` multiple times until I had gotten everything. I was quite surprised to find that other people had written stuff that did not do this, and even more surprised to find that it actually worked.

Editorial note: This example is flawed (the terminal drivers are implementation dependent and therefore do not necessarily behave as Gersten claims).

— Kevin Baranski-Walker

Publications Available

The following publications are available from the Association Office or the source indicated. Prices and overseas postage charges are per copy. California residents please add applicable sales tax. Payments **must** be enclosed with the order and **must** be in US dollars payable on a US bank.

USENIX Conference and Workshop Proceedings

Meeting	Location	Date	Price	Overseas Mail Air	Overseas Mail Surface	Source
USENIX	Wash. DC	Winter '87	$20	$25	$5	USENIX
Graphics Workshop III	Monterey	December '86	$10	$15	$5	USENIX
USENIX	Atlanta	Summer '86	$25	$25	$5	USENIX
USENIX	Denver	Winter '86	$20	$25	$5	USENIX
Graphics Workshop II	Monterey	December '85	$ 3	$ 7	$5	USENIX
USENIX	Portland	Summer '85	$25	$25	$5	USENIX
USENIX	Dallas	Winter '85	$20	$25	$5	USENIX
Graphics Workshop I	Monterey	December '84	$ 3	$ 7	$5	USENIX
USENIX	Salt Lake	Summer '84	$25	$25	$5	USENIX
UniForum	Wash. DC	Winter '84	$30	$20		/usr/group

EUUG Publications

The following EUUG publications may be ordered from the USENIX Association office.

The EUUG Newsletter, which is published four times a year, is available for $4 per copy or $16 for a full-year subscription. The earliest issue available is Volume 3, Number 4 (Winter 1983).

The July 1983 edition of the EUUG Micros Catalog is available for $8 per copy.

≡ ≡ ≡

Software Tapes

The 86.2 Software Distribution Tapes were all mailed prior to the end of February. If you have not received the second 1986 tape by the time this has been published, please contact *usenix!office* (415-528-UNIX).

It has been brought to the attention of the Association Office that the version of `Pathalias` on the 86.2 tape will not compile.

Peter Honeyman is working on a new version, which will soon appear in mod.sources.

— PHS

4.3BSD UNIX Manuals

The USENIX Association is sponsoring production of the 4.3BSD UNIX Manuals for its Institutional and Supporting members.† This article provides information on the contents, cost, and ordering of the manuals.

The 4.3BSD manual sets are significantly different from the 4.2BSD edition. Changes include many additional documents, better quality of reproductions, as well as a new and extensive index. All manuals are printed in a photo-reduced 6″×9″ format with individually colored and labeled plastic "GBC" bindings. All documents and manual pages have been freshly typeset and all manuals have "bleed tabs" and page headers and numbers to aid in the location of individual documents and manual sections.

A new Master Index has been created. It contains cross-references to all documents and manual pages contained within the other six volumes. The index was prepared with the aid of an "intelligent" automated indexing program from Thinking Machines Corp. along with considerable human intervention from Mark Seiden. Key words, phrases and concepts are referenced by abbreviated document name and page number.

While two of the manual sets contain three separate volumes, you may only order complete sets.

The costs shown below do not include applicable taxes or handling and shipping from the publisher in New Jersey, which will depend on the quantity ordered and the distance shipped. Those charges will be billed by the publisher (Howard Press).

Manuals are available now. To order, return a completed "4.3BSD Manual Reproduction Authorization and Order Form" to the USENIX office along with a check or purchase order for the cost of the manuals. You **must** be a USENIX Association Institutional or Supporting member. Checks and purchase orders should be made out to Howard Press. Orders will be forwarded to the publisher after license verification has been completed, and the manuals will be shipped to you directly from the publisher.

Manual	Cost*
User's Manual Set (3 volumes) User's Reference Manual User's Supplementary Documents Master Index	$25.00/set
Programmer's Manual Set (3 volumes) Programmer's Reference Manual Programmer's Supplementary Documents, Volume 1 Programmer's Supplementary Documents, Volume 2	$25.00/set
System Manager's Manual (1 volume)	$10.00

* Not including postage and handling or applicable taxes.

4.2BSD Manuals are No Longer Available

† Tom Ferrin of the University of California at San Francisco, a former member of the Board of Directors of the USENIX Association, has overseen the production of the 4.2 and 4.3BSD manuals.

4.3BSD Manual Reproduction Authorization and Order Form

This page may be duplicated for use as an order form

USENIX Member No.: _____

Purchase Order No.: _____

Date: _____

As the representative of a USENIX Association Institutional or Supporting Member in good standing, and as a *bona fide* license holder of both a 4.3BSD software license from the Regents of the University of California and a UNIX®/32V or System III or System V license or sublicense from AT&T and pursuant to the copyright notice as found on the rear of the cover page of the UNIX®/32V Programmer's Manual stating that

> "Holders of a UNIX®/32V software license are permitted to copy this document, or any portion of it, as necessary for licensed use of the software, provided this copyright notice and statement of permission are included."

I hereby appoint the USENIX Association as my agent, to act on my behalf to duplicate and provide me with such copies of the Berkeley 4.3BSD Manuals as I may request.

Signed (authorized Institutional representative): _____

Institution: _____

Ship to:

Name: _____

_____

_____

_____

_____

Phone: _____

Billing address, if different:

Name: _____

_____

_____

_____

_____

Phone: _____

The prices below **do not** include shipping and handling charges or state or local taxes. All payments must be in US dollars drawn on a US bank.

4.3BSD User's Manual Set (3 vols.)	_____	at $25.00 each	= $_____
4.3BSD Programmer's Manual Set (3 vols.)	_____	at $25.00 each	= $_____
4.3BSD System Manager's Manual (1 vol.)	_____	at $10.00 each	= $_____
Total	_____		$_____

[ ] Purchase order enclosed; invoice required.
(Purchase orders **must** be enclosed with this order form.)

[ ] Check enclosed for the manuals: $_____
(Howard Press will send an invoice for the shipping and handling charges and applicable taxes.)

Make your check or purchase order out to Howard Press and mail it with this order form to:

Howard Press
c/o USENIX Association
P.O. Box 7
El Cerrito, CA 94530

for office use: l.v.: _____ check no.: _____ amt. rec'd: _____

Local User Groups

The USENIX Association will support local user groups in the following ways:

• Assisting the formation of a local user group by doing an initial mailing for the group. This mailing may consist of a list supplied by the group, or may be derived from the USENIX membership list for the geographical area involved. At least one member of the organizing group must be a current member of the USENIX Association. Membership in the group must be open to the public.

• Publishing information on local user groups in *;login:* giving the name, address, phone number, net address, time and location of meetings, etc. Announcements of special events are welcome; send them to the editor at the USENIX office.

Please contact the USENIX office if you need assistance in either of the above matters. Our current list of local groups follows.

In the **Atlanta** area there is a group for people with interest in UNIX or UNIX-like systems, which meets on the first Monday of each month in White Hall, Emory University.

Atlanta UNIX Users Group
P.O. Box 12241
Atlanta, GA 30355-2241

Marc Merlin (404) 442-4772
Mark Landry (404) 365-8108

In the **Boulder**, Colorado area a group meets about every two months at different sites for informal discussions.

Front Range Users Group
USENIX Association Exhibit Office
Oak Bay Building
4750 Table Mesa Drive
Boulder, CO 80303

John L. Donnelly (303) 499-2600
usenix!johnd

A UNIX users group has formed in the **Coral Springs**, Florida, area. For information, contact:

S. Shaw McQuinn (305) 344-8686
8557 W. Sample Road
Coral Springs, FL 33065

Dallas / Fort Worth UNIX User's Group
Seny Systems, Inc.
5327 N. Central, #320
Dallas, TX 75205

Jim Hummel (214) 522-2324

In **East Lansing**, Michigan, the Michigan State University UNIX Users Group meets approximately monthly. It is open to all interested persons, not only University affiliates. For meeting times and more information, contact:

John H. Lawitzke (517) 355-3769
Division of Engineering Research
364 Engineering Building
Michigan State University
East Lansing, MI 48824

ihnp4!msudoc!eecae!lawitzke

In **Fresno,** California, a Central California UNIX User's Group is now being formed to bring together UNIX users, programmers, and administrators. Initially the group will consist of an electronic mailing list to which questions, comments, answers, rumors, and tips will be posted. Communication will be by uucp.

Educational and governmental institutions contact:

Brent Auernheimer (209) 294-4373
Department of Computer Science
California State University
Fresno, CA 93740-0109

CSNET: brent@CSUFresno.edu
uucp: csufres!brent

Commercial institutions, contact:

Gordon Crumal (209) 435-4242
Harry J. Wilson & Company
Insurance Correspondents, Inc.
2350 W. Shaw Ave, Suite 110
Fresno, CA 93711

uucp: csufres!tower!gordon

The **Los Angeles** UNIX Group (LAUG) meets on the third Thursday of each month in Redondo Beach, California. For additional information, please contact:

Drew Bullard (213) 535-1980
{ucbvax,ihnp4}!trwrb!bullard

Marc Ries (213) 535-1980
{decvax,sdcrdcf}!trwrb!ries

In **Minnesota** a group meets on the first Wednesday of each month. For information, contact:

UNIX Users of Minnesota
Shane McCarron (612) 786-1496
ihnp4!meccts!ahby
ahby@mecc.com

In the northern **New England** area is a group that meets monthly at different sites. Contact one of the following for information:

Emily Bryant (603) 646-2999
Kiewit Computation Center
Dartmouth College
Hanover, NH 03755

decvax!dartvax!emilyb

David Marston (603) 883-3556
Daniel Webster College
University Drive
Nashua, NH 03063

In the **New York City** area there is a non-profit organization for users and vendors of products and services for UNIX systems.

Unigroup of New York
G.P.O. Box 1931
New York, NY 10116

Ed Taylor (212) 513-7777
{attunix,philabs}!pencom!taylor

The **New Zealand** group provides an annual Workshop and Exhibition and a regular newsletter to its members.

New Zealand UNIX Systems User Group
P.O. Box 13056
University of Waikato
Hamilton, New Zealand

An informal group has started in the **St. Louis** area:

St. Louis UNIX Users Group
Plus Five Computer Services
765 Westwood, 10A
Clayton, MO 63105

Eric Kiebler (314) 725-9492
ihnp4!plus5!sluug

In the **San Antonio** area the San Antonio UNIX Users (SATUU) meet twice each month with the second Wednesday being a dinner meeting and the third Wednesday being a "roving" meeting at a user site.

San Antonio UNIX Users
7950 Floyd Curl Dr. #102
San Antonio, TX 78229-3955

William T. Blessum, M.D. (512) 692-0977
ihnp4!petro!bles!wtb

In the **Seattle** area there is a group with over 150 members, a monthly newsletter, and a software exchange system. Meetings are held monthly.

Bill Campbell (206) 232-4164
Seattle UNIX Group Membership Information
6641 East Mercer Way
Mercer Island, WA 98040

uw-beaver!tikal!camco!bill

A UNIX/C language users group has been formed in **Tulsa**. For current information on meetings, etc. contact:

Pete Rourke
$USR
7340 East 25th Place
Tulsa, OK 74129

USENIX Association
P.O. Box 7
El Cerrito, CA 94530

First Class Mail

Summer USENIX Conference

MINIX: A UNIX Clone for the IBM PC

The DASH Project

Summary of USENIX Board Meetings

USENIX Financial Report

Change of Address Form

Please fill out and send the following form through the U.S. mail to the Association Office at the address above.

Name: _____ Member #: _____

OLD: _____

_____

_____

_____

NEW: _____

_____

_____

_____

Phone: _____

uucp: {decvax,ucbvax}! _____